Scaling up question-answering to linked data

Conference or Workshop Item

Scaling up Question-Answering to Linked Data

Vanessa Lopez[1], Andriy Nikolov[1], Marta Sabou[1], Victoria Uren[2], Enrico Motta[1], Mathieu d'Aquin[1]

[1] KMI. The Open University, MK76AA, UK {v.lopez, a.nikolov, r.m.sabou, e.motta, m.daquin}@open.ac.uk, [2] The University of Sheffield, S14DP, UK v.uren@dcs.shef.ac.uk

Abstract. Linked Data semantic sources, in particular DBpedia, can be used to answer many user queries. PowerAqua is an open multi-ontology Question Answering (QA) system for the Semantic Web (SW). However, the emergence of Linked Data, characterized by its openness, heterogeneity and scale, introduces a new dimension to the Semantic Web scenario, in which exploiting the relevant information to extract answers for Natural Language (NL) user queries is a major challenge. In this paper we discuss the issues and lessons learned from our experience of integrating PowerAqua as a front-end for DBpedia and a subset of Linked Data sources. As such, we go one step beyond the state of the art on end-users interfaces for Linked Data by introducing mapping and fusion techniques needed to translate a user query by means of multiple sources. Our first informal experiments probe whether, in fact, it is feasible to obtain answers to user queries by composing information across semantic sources and Linked Data, even in its current form, where the strength of Linked Data is more a by-product of its size than its quality. We believe our experiences can be extrapolated to a variety of end-user applications that wish to scale, open up, exploit and re-use what possibly is the greatest wealth of data about everything in the history of Artificial Intelligence.

Keywords: question answering, link data, fusion, semantic web, natural language.

1 Introduction

The SW has expanded rapidly, offering a wealth of semantic data that can be used for experimental purposes, for example to enhance keyword search technologies. A prominent example is the amount of web data in the Linked Data [2] cloud. It is possibly, the largest Knowledge Base (KB) about everything in the history of Artificial Intelligence. Till now, most KBs covered specific domains and were created by relatively small groups. Yet, this is starting to change and we are reaching the critical mass required to realize the vision of large scale, distributed SW, with real-world datasets, representing real community agreement, and leading to astonishing research possibilities that can exploit and reuse these freely available data. For instance, the DBpedia project [3] extracts structured information from Wikipedia. The DBpedia ontology describes 170 classes in a shallow subsumption hierarchy, and more than 900 properties[1]. DBpedia has a high degree of conceptual overlap with other datasets, and it is increasingly becoming the central interlinking hub. The web of data around DBpedia covers 4.7 billion pieces of information about domains such as geography, people, companies, films, music, genes, amphibians, books and publications.

Ultimately, by integrating and connecting data on the web, Linked Data, and DBpedia in particular, as stated in [3] "can be used to answer quite surprising queries about a wide range of topics". E.g., by failing to link the content one can obtain films directed by Francis Ford Coppola but not what actors have played in any of his movies. However, while back

[1] Plus a dataset of 8000 property types for which there is no formal ontology (as November 2009).

end technologies and semantic applications can be robust at the small or medium scale, they may not be suitable when applying them to a real-world scale of heterogeneous web data. In other words, while Linked Data datasets literally may contain the answers to millions of questions, locating and exploiting the relevant information to extract these answers from them is a major challenge. In fact, most tools analyzed in Section 2 only perform a shallow exploitation of these data. Thus, in this paper we analyze the practicability of this ambition from the end-user application side, by looking at the scalability issues when integrating our multi-ontology QA system, PowerAqua, which uses state of the art methods from computational linguistics, ontology mapping, and data fusion, with some of the large general purpose data offered by the Linked Data community: DBpedia, the BBC backstage data whose scope is TV broadcasts and music [6], umbel[2] and musicBrainz.

Kaufmann and Bernstein [5] demonstrated, via a usability experiment comparing four query interfaces to an ontology, that casual users preferred the interface that used full NL queries to those using keywords, partial sentences and a graphical interface. Furthermore, the intuition that it would be easier to obtain answers from structured data than open text had lead to much interest in open NL interfaces that build and query their own massive trusted comprehensive factual KBs about the world (e.g., commercial ventures such as Powerset, START, Wolfram Alpha or True Knowledge[3]). Hence, it is worth considering an open NL interface for the end user to locate and query the Linked Data content on the web.

PowerAqua [9] takes a NL query from the user and retrieves answers from heterogeneous semantic data repositories. In particular, PowerAqua is able to integrate, on the fly, statements drawn from different sources to generate integrated answers to questions. Knowledge can be aggregated to complete information partially presented in single sources, fusing similar answers and filtering irrelevant ones. Furthermore, the most accurate answer(s), in terms of their relevance to the query and the varying levels of quality, popularity and trust, are elicited from different sources [8]. As such, PowerAqua supports users in locating, reusing and querying the open SW or organizations with large semantics intranets, where the information is distributed across independent departmental sites and external semantic sources. However, the SW has been rapidly evolving during the development of PowerAqua. PowerAqua was first envisioned in 2006 as a shift between the first generation of closed domain semantic systems, akin to smart KB applications, and the next generation of open SW applications to exploit the increasing amounts of semantic markup data, which is heterogeneous with respect to both the ontology characterization and provenance. Again, now, we can distinguish a new turning point in the evolution of the SW driven by the emergence of Linked Data. Querying Linked Data brings up a new scenario, the differentiating characteristics of which (detailed in Section 4) are:

I. Scalability is not only in the number of ontologies but also on their size (number of ontological elements). E.g., more than 2.9 million things are described in DBpedia.

II. From specific domain ontologies to large generic ontologies about everything, with a wider coverage of relationships across entities from a variety of domains.

III. Ontologies are decentralized, containing redundant and heterogeneous terminology, and connected to each other creating a network or cloud of ontologies.

In what follows, we look at the abilities of existing tools that handle the sheer amount of multi-domain data offered by Linked Data to provide easy access to the end user (Section 2). Then we briefly describe PowerAqua (Section 3), and present the major issues (Section

[2] Derived form *OpenCyc* and which consists of 20,000 classes (http://www.umbel.org/backbone.html)

[3] http://www.powerset.com/, http://start.csail.mit.edu/, http://www.wolframalpha.com/index.html and http://www.trueknowledge.com/ respectively.

4) that we faced to scale up PowerAqua to take advantage of Linked Data's potential to answer queries. The feasibility of the solutions presented (in Section 5) is assessed through initial experiments that measure the QA performance before and after using the main representative Linked Data set, DBpedia (Section 6). We finish by drawing some conclusions (Section 7). We believe that the lessons learned obtained with our experiments can be extrapolated to a large proportion of semantic tools that wish to retrieve, use and combine these large, rich multi-domain semantic data on the fly.

2 Motivations: Current interfaces for Linked Data and limitations

The database and SW communities had developed back-end technologies for managing large amounts of web data. Various RDF stores can scale over large amounts of data originating from different sources, such as Virtuoso or the Talis platform[4]. Search engines such as Watson [10] and Sindice [11] come also with features for indexing data from the SW. Linked Data sources usually offer a SPARQL endpoint for their dataset(s)[5]. Alternatively, they also provide RDF data dumps to build and query your own store[6]. However, users can hardly know which identifiers and properties are used in the KBs and hence can be used for querying. Consequently, they have to be guided when building queries, e.g., through the suggestion of reasonable alternatives. Creating innovative ways to interact with Linked Data is crucial and even envisioned as a potential "killer app".

Nonetheless, to find a trade-off between the complexity of the querying process and the amount of data it can use and integrate is still an open problem for ontology-based approaches. Semantic search models that have proved to work well in specific domains still have to undertake further steps towards an effective deployment on a decentralized, heterogeneous and massive repository of content about a potentially unlimited number of domains. Here we present a state of the art of the available user interfaces that can, in principle, scale enough to explore the Linked Data.

- Triple query builder interfaces: a Query Builder allows users to query the KB by means of multiple triple patterns. For each triple pattern variable, identifiers or filters for the subject, predicate and object can be defined. The user needs to follow the terminology and structure of the ontology to pose queries, e.g., the DBpedia Leipzig query builder [1]. However, for each typed identifier name a look ahead search proposes suitable options in a (in some cases long) drop down menu that helps the user to create complex queries, e.g.: <?x, rdf:type, db-ont:Person> <?x, notablePrize, Nobel_Peace_Prize>. Graph-based visualizations, on the contrary, in which all property values of the selected instances are analyzed for facet-filtering, are more resource demanding [1].
- Relationship finder: i.e., the DBpedia relationship finder [7] explores connections between objects. DBpedia is treated as an undirected graph and given two objects, the relationship finder looks for a path between them (not necessarily the shortest).
- Keyword lookup: e.g., the DBpedia URI Lookup Index and OpenLink Software[7] find the most likely matches (URIs) for a given term. The service combines Lucene's string similarity based ranking with a relevance ranking similar to PageRank. The OpenLink Software builds a text, label and URI lookup service upon a larger collection of sources, but it limits the number of results to only those containing an exact mapping of the input

[4] Virtuoso: http://virtuoso.openlinksw.com and Talis: http://www.talis.com/platform/
[5] A more complete list of SPARQL Endpoints at: http://esw.w3.org/topic/SparqlEndpoints
[6] Jena http://jena.hpl.hp.com/wiki/TDB; Sesame http://www.openrdf.org; 4store http://4store.org;
[7] http://lookup.dbpedia.org/ and http://lod.openlinksw.com

keyword and the search can be refined by specifying URIs of classes, properties or values, but usability is limited (e.g., 1358 classes are associated to the keyword "actor").

- Data aggregators and mash-ups: e.g., in Sig.ma (http://sig.ma/) the user enters a keyword and is able to explore all the aggregated data coming from the search engine Sindice (including synonyms and approximate mappings). Although mash-up technologies provide support for large-scale indexing and aggregating heterogeneous information, they do not attempt to disambiguate or rank between different interpretations, however, in Sig.ma the user can filter out the irrelevant sources.
- Linked Data browsers: they bring a way to browse RDF data on the web (data should be in RDF/XML or embedded in web documents using microformats). Examples are Tabulator and Disco[8]. Given a dereferenceable URI (i.e. an application can look up a URI over the HTTP protocol), these browsers render all information that they can find about that URI. All aggregated data across sources is viewed in a tabular form and the user can navigate through interlinked sources. However, it does not aggregate unconnected entities with different URIs representing the same individual.
- DBpedia Faceted search[9]: it allows the user to easily ask queries like "recent films about Buenos Aires", by typing the keyword "Buenos Aires" and then applying an intelligent filtering base on the underlying ontology, in this case by the item type "Film". The interface guides the user to filter objects according to facets (properties) and range of values. Nevertheless, the user needs to familiarize herself to some extent with the vocabulary and the structure of the KB, e.g., lexically related words like "Movie" are not understood. This applies not only to undefined terms, but also when there is not a straight mapping between the user query and the way the knowledge is structured in the ontology. For example, the query "Give me the husbands of Elizabeth Taylor" cannot be easily formulated if the user does not know that the relevant relation "spouse" is defined only for the entities representing Elizabeth Taylor's husbands, whose ontological type is "Actor", and it is neither defined for the instance "Elizabeth Taylor", nor for the types "Person" or "Artist". As the system is unable to map "husband" to the relation "spouse", the only solution left is to manually explore among all the results that contain a match for "Elizabeth Taylor".

Research on user interfaces for Linked Data is a open-ended area, each of the above paradigms have different abilities to handle the sheer amount of multi-domain data and the following limitations according to the criteria presented in Table 1:

1) Usability: if the input of the system is a URI or is some of the components of a triple, usability is limited to knowledgeable users, familiar with semantic source contents.
2) Expressivity: if the system builds upon keywords, then it provides limited capabilities to grasp and exploit the conceptualizations involved in user needs, with limitations such as the inability to account for relations between terms, or to cope with complex queries.
3) Scalability: if the system is restricted to one or a set of domains to maintain performance then it cannot scale to a truly open environment.
4) Mapping: the vocabulary that the system can understand is limited to that used in the ontology, the input is controlled and if a term has more than one sense, disambiguation is done manually by the user. Although these guided interfaces solve the old habitability problem (a mismatch between the user expectations and the abilities of the system), the burden or responsibility to formulate the queries is shifted from the system to the user.

[8] http://www.w3.org/2005/ajar/tab and http://sites.wiwiss.fu-berlin.de/suhl/bizer/ng4j/disco/
[9] http://dbpedia.neofonie.de/browse

5) Fusion: if the system has limited abilities to merge heterogeneous facts or multiple different answers (representing the same individual) across different semantic sources.
6) Ranking: the lack of ranking algorithms to cope with large-scale information sources.

Table 1. Limitations of the existent paradigms according to 6 criteria.

Criteria	Query Builder	Rel. Finder	Term Lookup	Mash-ups	Browsers	Facets
Usability	√	√	√	√	--	√
Expressivity	√	--	✗	✗	✗	√
Scalabilty	√	√	√	√	√	√
Mapping	✗	✗	--	--	✗	✗
Fusion	✗	✗	✗	--	--	✗
Ranking	✗	✗	--	✗	✗	✗

The existing querying approaches for Linked Data are restricted, among all of them, only facets and query builder interfaces provide an efficient way to pose complex and expressive queries to a large repository, with varying levels of usability depending on the query complexity and the number of filtered options presented on the drop menus, as end-users can get lost in large-scale information spaces. In fact, a common drawback of all these systems is that the user, not the application, is the one who has the responsibility and burden to reformulate the query in a way that can be understood. Another open issue concerns the usability of menu views and facets over multiple heterogeneous sources. Only keyword-based mash-ups (and lookup services to some extent) can aggregate information across sources, but they do not attempt to fuse similar results, nor do they have enough context to interpret and elicit the best answers. What is achievable on small/medium scale data by querying interfaces (in particular sophisticated NL ones) has until now not been achieved on large Linked Data. In this paper, we aim to go one step beyond the state of the art, by adapting the mapping and fusion techniques required by a particular NL interface, PowerAqua, to the Linked Data scenario.

3 PowerAqua: an open NL interface over structured data

Contrary to existing ontology-based NLI systems (see [5] for an example) whose scope is limited to one or a set of a-priori selected medium size ontologies at a time, PowerAqua dynamically selects and combines information drawn from multiple and heterogeneous ontologies in order to interpret and answer a query, making it worth investigating whether it can be successfully used to exploit the metadata offered by Linked Data. We briefly explain PowerAqua through the illustrative example "Find me university cities in Japan", a linguistically simple query that nevertheless, to be answered requires knowledge fusion across different sources. PowerAqua follows a pipeline architecture, the query is first transformed by the linguistic component[10] into a triple based intermediate format called Query-Triples (QTs): <university cities, ?, Japan>. At the next step, the QTs are passed on to the PowerMap mapping component, which identifies potentially suitable semantic entities in various ontologies that are likely to describe QT terms and answer a query, producing initial element level mappings. For example, PowerMap selects 81 mappings over 23 ontologies for the query term "Japan". PowerMap maximizes recall by searching for approximate mappings (e.g., "CountryJapan" in the TAP ontology) as well as exact mappings. These are jointly referred to as equivalent mappings. PowerMap also uses both WordNet and the SW itself (*owl:sameAs)* to find synonyms, hypernyms, derived words,

[10] The linguistic component uses GATE (www.gate.ac.uk) to transform into triples factual queries formed with wh-terms (which, what, who, when, where) or commands (give, list, show, tell,..).

meronyms and hyponyms (e.g., “Japanese islands”, “Nippon”, etc). In the third step, the Triple Similarity Service (TSS), exploring the relations of these entities, matches the QTs to ontological expressions or ontology based triple patterns specific to each of the considered semantic sources, producing a set of Onto-Triples (OTs), from which answers are derived as a list of entities matching the given triple patterns in each semantic source.

The algorithm iteratively refines candidates only as needed, analyzing most likely solutions first, and using more expensive but broader techniques last, if the previous steps fail to obtain a solution. First, the TSS chooses, whenever possible, the ontologies that better cover the user query and domain (i.e., it first searches for OTs in the ontologies that have potential mappings for at least two of the terms in a given QT). In our example, as PowerMap does not find any covering ontology with mappings for both arguments in the QT: “university cities” and “Japan”, the TSS algorithm reiterates again by splitting the compound term “university cities”, and consequently modifying the QT into: <cities / universities, ?, Japan> and creating a new QT for the compound <university, ?, cities>. For the QTs obtained in this second iteration, the TSS extracts, by analyzing the ontology relations, a small set of ontologies containing the valid OTs that jointly cover the user query and produce an answer. The resultant OTs are partially presented in Table 2.

Table 2. Triple Mapping Tables returned by PowerAqua for the example query.

QT_1: <cities / universities, ?, Japan>	
Agrovoc	OT_1 <*city, generic-location, Japan*> 27 answers: Ibaraki, Kyoto, kushiro-shi, etc.
SWETO	OT_1 <*city, attribute-country, Japan*> 30 answers: Chuo-ku Tokyo, Chuo-ku Osaka, etc.
KIM	OT_1 <*city, locatedIn, Country_TJA (Japan)*> 290 answers: Mishima, Kodaira, Soka, etc.
TAP	OT_1 <*city, locatedIn, CountryJapan*> 30 answers: CityChitose, CityKyoto, etc.
DBpedia	OT_1 <*RadioStation, city, Japan*> 2 answers: FM802, J-Wave.
QT_2: <university, ?, cities>	
DBpedia	OT_3 <*PopulatedPlace, city, University*> 7177 answers: Madrid (Polytechnic_University_of_Madrid), Kyoto (Kyoto_University), etc.

Finally, because each resultant OT only leads to partial answers, they need to be combined into a complete answer. The fourth component merges and ranks the various interpretations produced in different ontologies. In our example, 19 final answers are selected (e.g.: Kyoto (Kyoto_University), Fukushima (Fukushima_University)), by intersecting the answers from both QTs to obtain as a final set of answers those shared between the 7177 answers of *cities with a university* from DBpedia, and the more than 500 answers about *cities in Japan* derived from 8 ontologies (agrovoc, KIM, TAP, ATO, DBpedia, nepomuk, mid-level, and SWETO). Among other things, merging requires the system to identify similar entities across ontologies, e.g., “Kyoto” in DBpedia and “CityKyoto” in TAP. Furthermore, ranking (based on the quality and popularity of the mappings and answers) can be applied to sort the answers. E.g., a ranking measure is capable of providing a lower confidence to the noisy answers derived from the DBpedia OT: <*RadioStation, city, Japan*>. Although, in this case all answers derived after fusion are correct (as irrelevant answers only appear in one triple). As we show in [8], merging and ranking algorithms enhance the quality of the results.

To scale our model to a truly open web environment implies exploiting all the increasingly available semantic metadata in order to provide a good coverage of topics. To address this, PowerAqua is coupled with: a) the Watson SW gateway [10], which collects and provides fast access to the increasing amount of online available semantic data, and b) its own internal mechanism to index and query selected online ontological stores[11], as an alternative way to manage large repositories, like those offered by the Linked Data

[11] Using Lucene indexes and a common API to access and query ontologies (currently in sesame).

community, often not available in Watson due to their size and format (RDF dumps available for download as compressed files). One of the main issues for PowerAqua is to keep real time performance in a scenario of perpetual change and growth. The time needed to answer a query depends on: (1) the number of calls required to the Watson API or to query the repositories and the indexes, and (2) the response times to these calls. The total number of calls depends directly on the number of semantic sources and mappings that take part in the answering process. The response times to these calls depend on the complexity of the (SPARQL-alike) query and the size of the ontology. PowerAqua algorithms are optimized to reduce the number of expensive calls in the following way:

- The algorithm operates in a sequential fashion, by first collecting all candidate ontological entities for the terms in a query, and then identifying relevant relationships between them. However, the algorithm can re-iterate through the two different phases in order to look only for the mappings needed in the first instance. As such, the algorithm looks for mappings for terms forming a compound, e.g., "university cities", only if the first iteration, without splitting the compound, fails to produce results.
- The time consuming process of analyzing indirect relationships between two entities (i.e., relationships which require two triples to be joined, as the length of the path is limited to one mediating concept) is only carried out when no satisfactory is-a or ad-hoc direct relationships between any of the candidate entities within the ontology are found.
- The number of mappings returned by Watson is reduced through a functionality used to restrict the mappings for a given term to the ontologies that also contain mappings for another given term. E.g., in our query PowerMap retrieves hits for "university", "cities", or any of its lexical variations, from ontologies that also contain mappings for "Japan".

4. Before and After Linked Data: a new dimension in QA

We have identified three crucial factors that give a new perspective and draw a new line in the scope of SW applications before and after Linked Data, therefore, challenging Linked Data's potential in the QA context.

4.1 Scaling to highly populated and dense ontologies

A well-known limitation of the SW is its sparseness, as stated in [13], without a well populated SW, developing semantic search systems is only an intellectual exercise. Only a reduced number of topics were covered entirely or partially by an existing ontology (domain sparseness), and added to this, sparseness at the level of instances and relation (model complexity) was also found [4, 10]. However, Linked Data initiatives are producing a critical mass of semantic data, and it is likely that soon we will have so much data that the core issues would not be so much related to sparseness as to scalability and robustness.

Furthermore, as reported in an analysis of 25,500 ontologies and semantic documents collected by Watson [10] in 2007: "the SW is characterized by a large number of small, lightweight ontologies and a small number of large-scale, heavyweight ontologies", the biggest one at that time containing more than 28,000 entities. By contrast, an obvious characteristic of Linked Data is that it is very large, DBpedia alone consists of more than 103 million RDF triples, and describes more than 2.9 million entities. Also, as analyzed in [7] the DBpedia dataset is densely connected. As ever, the links between entities are more important than the entities themselves because that's where the context lives. Exploring

these connections between entities, e.g., through SPARQL queries, is the major bottleneck. Scale matters since the response time of the calls to query the ontologies depends directly on the size of the ontology, in particular when a query involves one or more classes with lots of instantiated data. Therefore, to query Linked Data mapping algorithms need to be able to explore the relevant connections while trying to avoid expensive computations.

4.2. Mapping query terms to large generic ontologies about everything

The really challenging aspects of these Link Datasets appeared to be not only their scale but also their heterogeneity: new challenges are introduced by simultaneously querying not only a large number of domain specific ontologies but also a few very large populated ontologies about everything.

The first version of PowerAqua was developed to work on an initial, sparsely populated SW. Therefore, PowerAqua algorithms were designed to maximize recall in order to bridge the gap between user terminology and terms used in the ontologies. Nevertheless, to keep up with the continuous and rapidly growth of the SW and to avoid analyzing an unfeasibly large space of solutions, the algorithms are iterative, and try to find an answer by augmenting the search space in each re-iteration until either an answer is found or all possibilities have been analyzed. The algorithm is based on the assumption that the ontologies that better cover a query (i.e. contain matches for most of the elements on a QT) are likely to represent better the domain(s) of the query and contain an answer. The algorithm uses this coverage criterion to restrict the mappings to be analyzed to those in the covering ontologies, extending the search space only if no answers are found.

However, the main distinctive feature of DBpedia, apart from its size, is that it is a repository about everything. Therefore, even in the cases where the answer to the user query is not contained in DBpedia, this dataset is frequently selected as relevant, and often contains a huge number of potential ontological hits, from a large number of domains, for one or more of the terms in the user query. Consequently, this coverage criterion to filter ontologies, ergo candidate mappings, becomes insufficient when most of the lexical matches for a user query belong to just one large open domain KB and often have different meanings than the one intended by the user. For instance, DBpedia alone contains more than 1000 mappings for an apparently unambiguous keyword like "Russian*":* e.g., the exact mappings Russia (instance) & russia (property); the synonyms Soviet_Union (instance) & USSR (instance); the hypernym country (class); the approximate instances: Russian_empire, president_of_russia, MTV_Russia, Rocket_to_Russia, Russia_Today, Anastasia_of_Rusia, etc. Analyzing the ontological context of all potentially relevant hits, to select the ones containing an answer, would result in unacceptably slow response time for a run time algorithm. Therefore, new filtering heuristics are needed.

4.3 Fusion across the heterogeneous and decentralized cloud of ontologies

When searching multiple collections together, knowledge needs to be shared and reused through fusion techniques. Redundant information or partial results from different sources need to be either combined and merged together or ranked in terms of their relevance to the query and the confidence in the answers derived from different sources.

Fusion requires matching at the schema level as well as entity reconciliation at the data level; it assigns the individuals returned as answers from different ontologies into subsets of answers that represent identical entities. A decision about the equivalence of two answers is made based on string similarity metrics applied to their labels, local names, and, in case of

uncertainty, other datatype attributes. Pairwise comparison of entities would make the complexity of the procedure N^2 with respect to the input set size. In order to avoid this, candidate matches are selected using a search over the indexes and the comparison focuses only on the entities that appear among the search results. This makes the complexity linear with respect to the answer set size, but in cases where the answer set is formed by thousands of partial answers, further heuristics to improve efficiency are needed.

5 Solutions to challenges and lessons learned

Next, we report on the solutions adopted by PowerAqua to solve the challenges outlined above. Our experiments also give us an insight on the quality of the datasets and a better understanding of the concrete issues encountered when handling large-scale data.

5.1 Large scale data: Shifting focus onto precision for mapping and fusion

DBpedia contains a large number of instances across domains, and as said before, it can produce a large number of diverse mappings for a single query term. Strategies to select the most likely mappings to answer a query and filter the least promising ones are crucial to keep run time performance especially when querying a huge amount of semantic data.

These strategies start with a quick filtering mechanism based on scores (not processing), returned by Lucene **string similarities**, to ensure that the mappings for a given term, within one single repository, have a minimum quality and their number does not go over a given threshold. This favors precision and can negatively affect recall, however, it is a necessary measure to ensure that all questions can be answered in real time. Secondly, the number of mappings is reduced according to **heuristics to filter out** the least promising individual mappings. These heuristics are not based on a PageRank-like algorithm (i.e., the popularity of the entity within the ontology)[12] but rather on the context of the query. The reason behind this is that we do not want to penalize searches in which the user is interested in the unique meaning (not the popular meaning) of the word. For instance, the meaning of Turkey is clear when asking for "Give me books written in Turkey" or "which wine is good for a meal based on Turkey?". Based on the **coverage criterion** ontologies about geography or food would be selected respectively. However, a dataset like DBpedia contains both meanings of Turkey (*Turkey_(bird)* and *Turkey*) plus several mappings for "books" and "wine". Before the time consuming process of analyzing the relationships between the mappings to obtain the interpretation that better translates the query in a given ontology, further heuristics to reduce the number of mappings within an ontology are applied. These heuristics are **based on the syntactic relevance** of the mappings: (1) the quality: exact vs. approximate; (2) the semantic relation: equivalent, synonyms, hyper(hypo)rnyms, or meronyms. Next, more expensive semantic mechanisms based on the **ontology taxonomy** are applied, i.e. to discard redundant mappings by selecting those that are higher in the same ontology hierarchy (e.g., the class "wine" selected over its subclasses "rose-wine", "white-wine", "sugary-wine", etc). Also, **unconnected mappings** with no ontological context, in the form of ad-hoc or is-a relations, are discarded.

These strategies applied by PowerMap mapping algorithms to increase precision, and consequently performance, require making certain assumptions about the quality of the semantic sources, without making any unreasonable or a-priori assumption. As explained in

[12] Popularity measures are used only to rank the final answers obtained across ontologies [8].

Section 5.2, if the heuristics are too strict recall is affected and valid answers are missed, in particular for the heterogeneous and general datasets such as DBpedia. The mapping finishes with, the TSS iterative algorithms [9] that exploit the **ontological relationships** between the candidate entities to translate the user query (starting with the most straightforward solutions and executing expensive steps last, if no solution is found).

In order to improve the efficiency of the fusion procedure, our approach is to balance the quality of the resulting set of the answers and the expected time cost. When the number of answers, which have to be processed by the fusion module is large, our procedure tries to minimize the number of index search calls. The input of the fusion procedure is a set of answers retrieved from one or more ontologies $o_i, i = 1..n$. Let A_i denote a set of answers $\{a_{i1}..a_{im}\}$ coming from the ontology o_i. By default, for each a_{ij} we searched the index using as keywords the label and local name of a_{ij} together with their synonyms extracted from WordNet. Instances a_{kx} which were retrieved by the search, and at the same time belonged to the original set of query answers, were considered as candidates for fusion. In the new version of the algorithm, we introduced two thresholds for $|A_i|$. The first regulates the use of WordNet synonyms in search: if $|A_i \rhd \lambda$, then WordNet synonyms are not used when searching for instances similar to $a_{ij} \in A_i$. The second one excludes the whole set A_i from search: if $|A_i \rhd \mu$, then the module does not try to search for instances similar to any $a_{ij} \in A_i$. Thus, if there is an instance a_{kx} belonging to an ontology o_k such that $a_{kx} \equiv a_{ij}$, they can only be merged if a search for a_{kx} returns a_{ij}, which potentially leads to lower recall of the fusion procedure. However, this potential loss of recall is justified when we are dealing with very large answer sets.

5.2 Heterogeneity and duplicated terms: filtering based on quality and semantics

In DBpedia, the most valuable contents extracted from Wikipedia are the *infoboxes*. However, different infobox templates use different names for the same attribute (e.g., *birthplace* and *placeofbirth*). The DBpedia project deals with this situation by using two different extraction approaches in parallel: a *generic* one that aims at a high coverage, and a *mapping-based* one that aims at high data quality by mapping Wikipedia terms to a manually created ontology [3]. However, the latest covers only 350 Wikipedia templates.

One of the filtering heuristics to favor precision is to consider the quality and semantic relation of the mappings. However, the presence of duplicated entities within the same semantic source limit the effectiveness of this criterion. Consider the example: “Give me the husbands of Elizabeth Taylor”, the keyword “husbands” produce several mappings in DBpedia, the approximate equivalent instances: Clifford_Husbands, Commuter_Husbands, Dead_Husbands, Husbands_and_Wives, Young_Husbands, etc; the exact equivalent properties: husbands, husband, etc; and the exact hypernyms: spouse, spouses, partner, etc. The equivalent properties representing “husband(s)” do not produce any valid OTs with the entity “Elizabeth Taylor”. The answer is encoded in the hypernym property “spouse”. Therefore, hypernyms can be as relevant as equivalent mappings. With this kind of dataset, in which different terms have the same semantics, heuristics need to be flexible. First, exact mappings, if any, independent of their semantic relation, are selected over approximate mappings. Furthermore, exact mappings are a requirement in cases where the type of the mapping is not the expected one. E.g., in “who won a Nobel prize” the linguistic relation “won” is mapped to the instance “Won_James_Won”, this approximate mapping is discarded before analyzing how it relates to the subject and object of the triple.

The TSS ultimately establishes the semantic validity of the candidate mappings by analyzing the ontological context, which is translated in many SPARQL-like queries to find

the OTs. However, for queries that are particularly expensive, heuristics based on the semantic relation (synonym, hyper(hypo)nym) together with the quality of the mapping (exact, approximate) are also applied to minimize the number of those expensive queries. In large ontologies like DBpedia, searching for both *ad-hoc relationships* (1:1) between two highly populated classes and *indirect relationships* with one mediating concept (1:2) are expensive computations. In these two cases, to favor precision when looking for relationships among the mappings, only pairs in which at least one candidate mapping is exact and equivalent are analyzed (singulars and plurals are considered exact mappings). In this way for the query "which languages are spoken in a country?" the TSS avoids searching for relations between DBpedia matches such as "text", an hypernym of "language", and "land", a synonym of "country".

Finally, heterogeneity is not only present in the vocabulary but also in the granularity of the data (i.e., entities modeled with different degrees of richness). For instance, the deepness that characterizes the YAGO hierarchy [12] and its conjunctive schema classes (used in DBpedia classification), which encode too much information in one class, e.g., "MultinationalCompaniesHeardquarteredInTheNetherlands", make the processing of the labels too difficult for automatic QA understanding.

5.3 Lack of semantics or formal ontology: light-weight reasoning

PowerAqua is able to scale thanks to the various strategies and filtering heuristics that keep the number of mappings and queries to the semantic sources more or less constant, even when adding large semantic sources or a large number of them. However, as said in Section 3, performance also depends on the size of the ontologies, which influences the response time of the calls (SPARQL or Serql) to query them. The effectiveness of these queries, which use the ontology semantics to perform basic light-weight inferences based on the taxonomy and relationships, also relates to the quality of the sources they are querying.

In DBpedia the properties defined in the namespace http://dbpedia.org/ontology belong to the data generated by the mapping-based approach. In this approach, fine-grained rules are applied to define the target datatype and ignore additional text that may be present in the attribute value. However, although the percentage of properties pointing to other DBpedia entities is much higher in the mapping-based dataset (53%) than in the generic dataset (25.6%) [3], the coverage is lower (843,000 compared to 1,462,00 entities). In the generic approach, http://dbpedia.org/property/, coverage of all infoboxes is complete but synonymous attribute names are not resolved, and there is a high error rate to determine the datatype of a value. The effectiveness of finding answers in the generic approach is limited with respect to the mapping approach and it has an important impact on query performance.

a) Domain and range: In the mapping approach OTs can be extracted with reasonable performance by querying the schema (domain and range). However the answer can be encoded in a property defined within the generic approach, where properties do not map to a schema. The lack of domain and range information, results in either sending expensive SPARQL / SeRQL queries or missing connections between the analyzed entities. To balance performance and recall, domain and range information is crucial in cases where we are looking for indirect relationships, or ad-hoc relations between two classes or a class and a literal. For example in "give me languages used in Islamic countries?" the query is translated into OTs representing "languages spoken in a country" and "countries that are Islamic", domain and range are used to get all possible relations among the two classes "language" and "country", because it is not feasible to check all the instances of languages to find ad-hoc relations with any instance of country in real time, particular for highly

populated classes. Therefore, the answers encoded in the OTs formed with the KB relation "http://dbpedia.org/property/states" are not found, but it finds answers for the schema relation: <language, http://dbpedia.org/ontology/states, country>. Furthermore, domain and range information is also needed in order to complete a triple when the relation is mapped but not the subject of the triple. E.g., the query "in which region is Cantonese spoken?" is mapped to the OT: <PopulatedPlace (domain of region), region, Cantonese> with 12 answers (Hong Kong, Macau, etc). Because an instance can instantiate a relation with thousands of other instances, schema information is needed to model the OT.

b) Inference of relationships: When looking for relationships between two entities, the ontology is treated as an indirect graph, and all direct and inverse relations are retrieved. Inheritance of relations is considered: if the ontological platform does not offer schema inferencing (as it is the case for Postgres repositories in Sesame 2), then complex *SeRQL* queries need to be generated to consider the relationships defined for the superclasses of the classes involved. In our previous example, domain and range information is used to find instantiated relations between the classes "language" and "country", or any of their superclasses. If looking for indirect relations inherency is also considered, but the search is restricted to candidate mediated concepts with relations defined in the schema, looking for indirect inverse relations is avoided as it is computationally expensive to search for relations to and from highly populated mediated concepts. Also, inferences cannot be done with instances whose type is defined in another ontology.

c) Literals and good enough labels: a literal has no structure and the meaning is given just by its label. E.g, the unprocessed value for the property "states" in the instance "Tamil language" ("*India, Sri Lanka and Singapore, where it has an official status; with significant minorities in Canada, {..}*") is too complex to automatically infer answers.

6 Initial Experiments and Discussion

Despite using community driven large scale knowledge obtained from sources that are heterogeneous, redundant and not always complete or well formed, the SW technology is mature enough to interpret and answer NL user queries. In this section, we present some example queries[13] to justify our claim that we can obtain answers to queries directly from the DBpedia semantically rich information – even in its current form. The solutions in Section 5 had allowed us to improve PowerAqua mapping and fusion algorithms to exhibit better performance, measured in terms of speed or seconds to answer a query, by shifting the focus on precision while minimizing the loss in recall. We have made an initial comparative study between PowerAqua efficiency before and after adding DBpedia dataset and the heuristics to favor precision, by using the semantic data and a subset of queries from previous evaluations [9, 8, 4]. This semantic data consists of 700 semantic documents distributed in 130 repositories, 3GBs data in which the biggest source (SWETO and SWETO-DBLP) is not more than 1GB (over 3 million triples). A sample of 16 queries can be seen in Table 3, where the last 6 queries can only be answered if DBpedia is included in the query dataset. As shown, the average time for the mapping algorithms to translate a query increases from 32 to 48 secs when using the same queries and datasets but adding DBpedia, even with the use of filtering heuristics. However, the resulting number of valid answers obtained after applying the fusion algorithm (which has a precision of 94%[15]) rises from 64 to 370 on average. The average mapping time with DBpedia increases to 52 secs when adding complex queries like Q_{11} and Q_{12} that require fusing partial translations

[13] More demo queries at: http://kmi.open.ac.uk/technologies/poweraqua

from DBpedia and other datasets to obtain a complete one. While this is acceptable for a research demo, work still has to be done to improve the speed. In any case, it shows that semantic data can be handled in modest projects, our demo runs in one reasonable sized server (a 3GHz Intel Pentium dual core with 8GB RAM).

The reasons behind the decrease in speed are not so much because of the increase of the number of resultant hits obtained when querying more and larger repositories. Heuristics that balance precision and recall reduce SeRQL calls by more than 40% (352/587) and even keep them lower (540) when mapping complex queries into multiple facts. However, speed falls because of a suboptimal performance at the back end, where the response times to calls to the repositories increases for single large datasets, in particular for expensive queries to find: (1) relationships between instances of highly populated classes (domain-range information is limited and inherency is take into account); (2) indirect relationships; (3) relationships involving literal values. The first fact explains that Q_1 is executed faster than Q_{12}, even if it implies twice as many (*SeRQL*) calls, because there are 47,821 actors starring in films in DBpedia while there are just 3,224 languages related to a country. The second and third facts explain why Q_9 is the slowest query, because answers are obtained in DBpedia through 21 indirect OTs, with mediating concepts such as airlines, company, person, military conflict, that relate to both DBpedia entities "island" and "spain", plus the SWETO ontology contains multiple literal values for "spain" (corresponding to instances of Spanish cities) that need to be analyzed. Nevertheless, we are optimistic that PowerAqua (and other query-intensive interfaces) can scale to a huge amount of semantic information, as long as the semantic software it is based on can efficiently respond to the growth of the semantic sources. Furthermore, if the quality of ontologies improves and more semantic data becomes available, it should be easier to find more precise mappings with answers.

Table 3.: Examples of queries after and before using the DBpedia dataset + heuristics (precision)

(Q_i) NL Query: After DBpedia / Before Dbpedia	**N°Ont**	**Answ.**	**Secs**	**Calls**
Q_1: How many languages are used in Islamic countries?	2/ 2	170/ 0	95.2/ 34.5	1078/ 419
Q_2: Which Russian rivers end in the Black Sea	3/ 1	4/ 1	41.3/ 27.3	639/ 428
Q_3: Who lives in the white house	4/ 3	12/12	17.9/ 13.7	310/ 144
Q_4: Give me airports in Canada	2/ 1	156/155	23/ 14.22	157/ 40
Q_5: List me Asian countries	6/ 6	64/ 72	15.3/ 67.4	298/ 1308
Q_6: Give me the main companies in India	2/ 2	710/ 386	17.4/ 43.9	298/ 588
Q_7: Give me movies starring Jennifer Aniston	3/ 2	28/ 23	10.7/ 4.5	94/ 22
Q_8: Which animals are reptiles?	9/ 8	2518/ 23	42.8/ 7.1	165/ 49
Q_9: Which islands belong to Spain	3/ 3	13/ 7	206/ 104	387/ 2617
Q_{10}: Find all the lakes in California	2/ 2	37/ 2	13.6/ 12.9	103/ 258
Average (10 queries) – after / before DBpedia	**3.6/3**	**371/68**	**48.3/32**	**352/587**
Q_{11}: Find me university cities in Japan	7/-	19/-	68/-	1087/-
Q_{12}: Tell me actors starring in films directed by Francis Ford Coppola	3/-	135/-	120/173	574/-
Q_{13}: Show me Spanish films with Carmen Maura	1/-	2/-	30.5./-	477/-
Q_{14}: Give me English actors that play in Titanic	1/-	4/-	144/-	3340/-
Q_{15}: Give me tennis players in France	1/-	29/-	14.7/-	113/-
Q_{16}: Television shows created by Walt Disney	1/ -	8/ -	9.4/ -	137/ -
Average (16 queries) – after DBpedia	**3.1**	**244.3**	**54.3**	**578.5**

Table 4.: Fusion with DBpedia – after precision heuristics / before precision heuristics

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	**Avg**
Secs	16.5/516	6.7/25	1.8/2	0.3/77	50.9/30	55.7/219	0.1/6.5	74.43/255	3.6/2.3	1.6/3	126/940	30.7/188
Calls	11/771	16/177	14/16	1/311	131/138	370/1148	44/53	19/2803	14/26	38/40	579/3792	112/843

We do not always have enough ontological context to focus on precision when, because of heterogeneity, there are many alternative translations. Take the query Q_{14}, which requires an unusual number of calls (3340) to find the OTs that translate to the QTs <actors/ English, play, Titanic> and <English, ?, actors>. There are 31 OTs in DBpedia for the first

QT linking the class "Actor" to 7 instances of "Titanic" (Cinematic_Titanic, Titanic_1943_film, Titanic_1953_film, Titanic_1997_film, Titanic_TV_miniseries, etc), through several ad-hoc relations (starring, director, producer, academyawards, etc.), because the matches for the linguistic relation "play" (the properties: play, plays and show) turn out not to be relevant for the query. Similarly, the second QT is mapped to 18 DBpedia OTs formed with various ad_hoc relations (residence, ethnicity, location, hometown, deathPlace, etc, and the duplicated properties: birthplace, birthPlace and born) between the class "Actor" and the instance "England". Moreover, the keyword "English" alone produces several mappings that had to be analyzed to determine or not their relevance (e.g., English language, English people, English channel, English actor, English football, etc).

Yet, the needed filtering heuristics impose a toll in recall, e.g., new DBpedia answers are obtained for Q_5 but the answers from one ontology (KIM) are missed, producing a loss in total answers (64/72). Moreover, in Q_{13}, relevant films are missed because the mapping "spanish_language" was not selected and only OTs formed with the mapping "Spain" were retrieved. Notwithstanding, generally, we cannot see any negative effect on recall if we compare the total number of final answers with the previous version of PowerAqua but quite the contrary (371/68), in fact, many of the missed mappings were imprecise (and in any case filtered after fusion), like the city "Mammoth Lake" as an answer for Q_{10}, or their answers were already obtained from other ontologies.

Table 4 compares the average fusion times before and after applying the fusion heuristics to reduce the number of calls to the indexes, and therefore improve efficiency. The average fusion times have been reduced from 188 secs to 30.7 secs. We could not see any loss in recall due to the new fusion heuristics in this query sample.

Nevertheless, there are many open grounds for exploration of techniques to: (1) lead to better mappings, e.g., "British actors in Titanic" translates into multiple OTs but with no answers after fusion, because British is not mapped to England, and the resultant answers like "Kate Winslet" are related to England but not to Britain; (2) improve the selection of mappings by trust mechanisms and user feedback; (3) we have not yet fully exploited the explicit linkage for bridging data in the Linked Data world, e.g., if no ontology offers a complete translation of a single QT, instead of obtaining partial translations, explicit connections stated across different sources (*owl: sameAs*) can be use to efficiently search for cross ontology connections across entities as if they were part of the same graph.

7 Conclusions and Future Work

In the early stages of development, sparseness overshadowed the potential for QA using semantic data [4], however the transition from restricted domains to real world scale structured datasets stimulated by Linked Data, adds a new dimension of scalability for both applications and back-end technologies that aim to exploit the SW, opening new QA possibilities, beyond prototypes and proof of concepts. In this paper, we analyzed the implications of fusion and mapping techniques for querying large scale Linked Data content across multiple domains in NL, which allow us to extract useful lessons for the Linked Data community and developers in the wider SW community.

Existing techniques focusing simply on effectiveness may not scale to large amounts of data. To scale to a large amount of data, applications need to leverage precision and recall needs with the potential of scaling up through parallelization and adaptive load balancing. Although still more work needs to be done in our back end infrastructure to cover not just a subset but all Linked Data available, in this paper we analyzed the implications from the front-end (application) perspective on the way the query process should be realized to

efficiently extract answers from large and highly heterogeneous community-driven open data, beyond any particular implementation, and in particular, the issues that we have addressed in order to scale our mapping and fusion techniques to millions of triples.

Currently, evaluation initiatives that would allow formal evaluations and direct comparison between systems are being investigated, e.g., in the SEALS (semantic evaluation at large scale) project. In this work our aim is to present some examples and initial experiments to justify our claim that we can obtain answers to queries from the huge amount of semantically rich information extracted from DBpedia – even in its current form.

QA over Linked Data opens the way to ambitious future research directions. For instance, DBpedia entities are also being used to annotate web content, i.e., DBpedia is being used as the controlled vocabulary to annotated BBC news. In this way, stories across different BBC domains are linked together through DBpedia data [6]. As the number of annotated sites increases, the answers to a question extracted by PowerAqua in the form of lists of entities (e.g., from DBpedia), that can be used as a valuable resource for discovering classic web content that is related (annotated) with those entities. Our future goal is to complement the answers given by PowerAqua with web pages to enhance the expressivity of traditional search engines with semantic information. Summing up, PowerAqua aims to provide a service that extends capabilities from querying a large number of unconnected sources to more interlinked ecosystems of data, in response to a user demand.

References

1. Auer, S., and Lehmann, J. What have Innsbruck and Leipzig in common? Extracting Semantics from Wiki Content. In Proc. of the 4th European Conference on the Semantic Web, Austria, 2007
2. Bizer, C., Heath, T., Berners-Lee, T. Linked Data – The Story So Far. Int. Journal on Semantic Web and Information Systems, Vol. 5(3), 2009.
3. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellman, S. DBpedia. A Crystallization Point for the Web of Data. Journal of Web Semantics, Vol. 7(3), 2009
4. Fernandez, M., Lopez, V., Sabou, M., Uren, V., Vallet, D., Motta, E., and Castells, P. Semantic Search meets the Web. In Proc. of the Int. conference on Semantic Computing, California, 2008
5. Kaufmann, E., Bernstein, A. - How Useful Are Natural Language Interfaces to the Semantic Web for Casual End-Users? In Proc. of the ISWC/ASWC, Korea, 2007
6. Kobilarov, G., et al. Media Meets Semantic Web --- How the BBC Uses DBpedia and Linked Data to Make Connections. In Proc. of the 6th European Semantic Web Conference, Greece, 2009
7. Lehmann, J., Schüppel, J., Auer, S. Discovering unknown connections – the DBpedia relationship finder. In Proc. of the 1st SABRE Conference on Social Semantic Web, Germany, 2007
8. Lopez, V., Nikolov, A., Fernandez, M., Sabou, M, Uren, V. and Motta, E. Merging and Ranking answers in the Semantic Web: The Wisdom of Crowds. In Proc. of the ASWC, China, 2009
9. Lopez, V., Sabou, M., Uren, V. and Motta, E. Cross-Ontology Question Answering on the Semantic Web – an initial evaluation. In Proc. of the KCAP Conference, California, USA, 2009
10. d'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M., Motta, E. Characterizing knowledge on the semantic web with Watson. In Proc. of 5th Int. EON Workshop, Korea, 2007
11. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G. Sindice.com: A document-oriented lookup index for open linked data. Journal of Metadata, Semantics and Ontologies, 3(1), 2008
12. Suchanek, F. M., Kasneci, G., Weikum, G. YAGO: A Large Ontology from Wikipedia and WordNet, Web Semantics. In Proc. of the WWW, 6 (3), 2008
13. Uren, V., Sabou, M., Motta, E., Fernandez, M., Lopez, V., Lei, Y.: Reflections on five years of evaluating semantic search systems. Journal of Metadata, Semantics and Ontologies, 5(2), 2010